Monte Carlo Sampling Methods Homework 3

Zihan (Steven) Zhang

October 15, 2022

I have consulted and discussed the materials with Qianyu Zhu.

1 Exercise 28

For notation, we have:

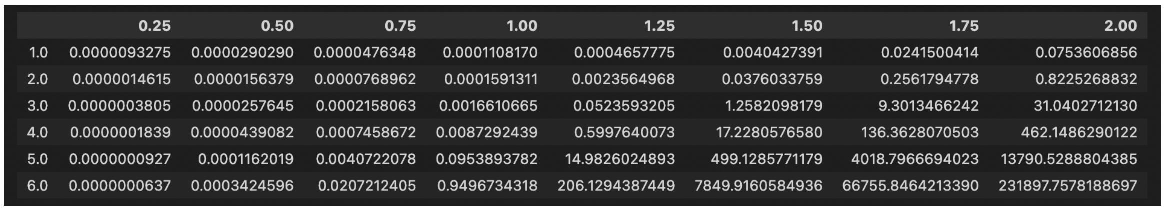
$$\pi = \mathcal{N}(0, \sigma^2) \quad \pi' = \mathcal{N}(0, 1)$$

$$\omega(x) = \frac{\pi}{\pi'} = \frac{1}{\sigma} e^{x^2/2 - x^2/2\sigma^2}$$

We would then implement the resampling methods (multinomial, Bernoulli, and systematic) respectively and use $\mathcal{N}(0, 1)$ random variables to generate approximate $\mathcal{N}(0, \sigma^2)$ random variables based on the weight $\omega(x)$.

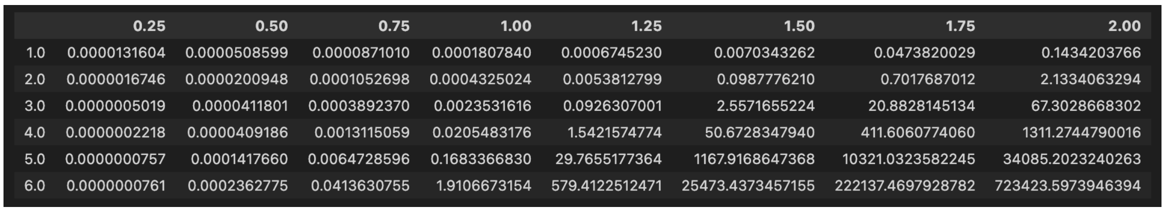
- Bernoulli: Referring to page 56 in lecture note: for each step, we have n samples $\{x_k\}_1^n$ and weight $\{\omega_k\}_1^n$; thus, we could choose $N_i = \lfloor n\omega_k \rfloor + 1_{u_k < n\omega_k - \lfloor n\omega_k \rfloor}$ for some independent random variable u_k drawn from $\mathcal{U}(0, 1)$; we could then create N_i samples equal to x_k in this new collection.
- Multinomial: Using the inversion method, we have: for each step, we have n samples $\{x_k\}_1^n$ and weight $\{\omega_k\}_1^n$; then we could draw the distribution $Multinomial(N, p)$ with the vector p having entries $p_i = \omega_i$; then we have the new samples $\{y_k\}_1^n$ with weight $\frac{1}{n}$.
- Systematic: Similar to before, we have: for each step, we have n samples $\{x_k\}_1^n$ and weight $\{\omega_k\}_1^n$; then we could generate a single independent random variate u from $\mathcal{U}(0, 1)$, then by page 57 in lecture note, $N_i = \left| \left\{ \sum_{i=1}^{j-1} \omega_i \leq \dfrac{m+u}{n} \leq \sum_{i=1}^{j} \omega_i, m \in \mathbb{N} \right\} \right|$ where $|A|$ for a discrete set of points A refers to the number of points in A; we could then create N_i samples equal to x_k in this new collection.

Here we also present the variance of important sampling estimator $N^{(k)}$ using these 3 methods (Figure 1, 2, 3) with different target density variance and target statistics. The x-axis corresponds to σ and y-axis corresponds to k. Noteworthy, when $\sigma < 1$, the estimator for higher moments (i.e. large k) would still have a lower variance, and it would quickly blow up when $\sigma > 1$. Qualitatively (and intuitively), there is a strong positive correlation between σ and the variance. For large σ, the weight $\frac{\pi}{\pi'}$ are big for samples with large absolute value, and the error would be aggregated in resampling. In terms of parallel comparison, when $\theta < 1$, the differences between the 3 methods are negligible. However, when $\sigma > 1$, for polynomial resampling, var$(N^k) = NW_n^k(1 - W_n^k)$, so the variance is order $\mathcal{O}(N)$ and is comparatively large. Meanwhile, for Bernoulli resampling, var$(N^k) = (\lceil NW_n^k \rceil - NW_n^k)(NW_n^k - \lfloor NW_n^k \rfloor)$, so the variance is order $\mathcal{O}(1)$ and is comparatively small. These theoretical arguments are supported by the results in Figure 1, 2, 3, which could be reproduced by running the attached code.



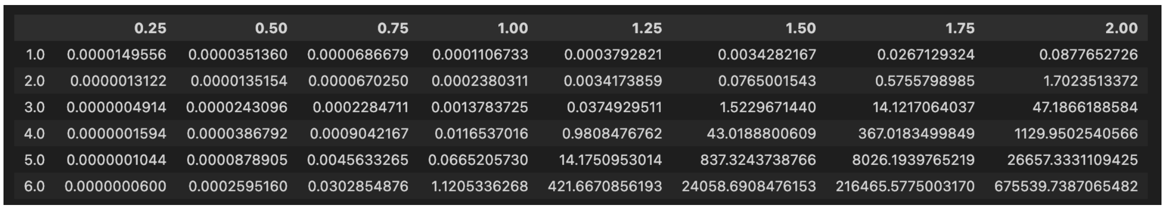
	0.25	0.50	0.75	1.00	1.25	1.50	1.75	2.00
1.0	0.0000093275	0.0000290290	0.0000476348	0.0001108170	0.0004657775	0.0040427391	0.0241500414	0.0753606856
2.0	0.0000014615	0.0000156379	0.0000768962	0.0001591311	0.0023564968	0.0376033759	0.2561794778	0.8225268832
3.0	0.0000003805	0.0000257645	0.0002158063	0.0016610665	0.0523593205	1.2582098179	9.3013466242	31.0402712130
4.0	0.0000001839	0.0000439082	0.0007458672	0.0087292439	0.5997640073	17.2280576580	136.3628070503	462.1486290122
5.0	0.0000000927	0.0001162019	0.0040722078	0.0953893782	14.9826024893	499.1285771179	4018.7966694023	13790.5288804385
6.0	0.0000000637	0.0003424596	0.0207212405	0.9496734318	206.1294387449	7849.9160584936	66755.8464213390	231897.7578188697

Figure 1: Using Bernoulli resampling method.



	0.25	0.50	0.75	1.00	1.25	1.50	1.75	2.00
1.0	0.0000131604	0.0000508599	0.0000871010	0.0001807840	0.0006745230	0.0070343262	0.0473820029	0.1434203766
2.0	0.0000016746	0.0000200948	0.0001052698	0.0004325024	0.0053812799	0.0987776210	0.7017687012	2.1334063294
3.0	0.0000005019	0.0000411801	0.0003892370	0.0023531616	0.0926307001	2.5571655224	20.8828145134	67.3028668302
4.0	0.0000002218	0.0000409186	0.0013115059	0.0205483176	1.5421574774	50.6728347940	411.6060774060	1311.2744790016
5.0	0.0000000757	0.0001417660	0.0064728596	0.1683366830	29.7655177364	1167.9168647368	10321.0323582245	34085.2023240263
6.0	0.0000000761	0.0002362775	0.0413630755	1.9106673154	579.4122512471	25473.4373457155	222137.4697928782	723423.5973946394

Figure 2: Using Multinomial resampling method.



	0.25	0.50	0.75	1.00	1.25	1.50	1.75	2.00
1.0	0.0000149556	0.0000351360	0.0000686679	0.0001106733	0.0003792821	0.0034282167	0.0267129324	0.0877652726
2.0	0.0000013122	0.0000135154	0.0000670250	0.0002380311	0.0034173859	0.0765001543	0.5755798985	1.7023513372
3.0	0.0000004914	0.0000243096	0.0002284711	0.0013783725	0.0374929511	1.5229671440	14.1217064037	47.1866188584
4.0	0.0000001594	0.0000386792	0.0009042167	0.0116537016	0.9808476762	43.0188800609	367.0183499849	1129.9502540566
5.0	0.0000001044	0.0000878905	0.0045633265	0.0665205730	14.1750953014	837.3243738766	8026.1939765219	26657.3331109425
6.0	0.0000000600	0.0002595160	0.0302854876	1.1205336268	421.6670856193	24058.6908476153	216465.5775003170	675539.7387065482

Figure 3: Using Systematic resampling method.

2 Exercise 29

For the SAW function, we input the length of walking D, the number of samples N, the length of lattice L, and the flag of whether or not to use systematic resampling (which would prevent degeneration). The return values are the independent self-avoid random walk with corresponding weights (in the vector form). We set the L value to be large so that the SAW is less possible to break through the boundary, though formulation is periodic. Figure 4 graphically shows the results under different D values, where the start and end points are highlighted. To validate the results, we check the expected number of times a lattice site is visited, which should be the same for every site. To approach this, we would generate 10000 SAW with length 100 on the 20×20 lattice and visualize the visiting frequency on each node. For every step of a valid sample, we add sample weight to its current position and normalize the matrix after the calculations. After we resample it, the large weight is recalculated and the color distribution becomes more even (observed from the color bar scale), as shown in Figure 5. For other statistics, we could observe the distribution of ending positions that might be helpful to validate and debug the codes. To compare the methods, we could compare the time consumption, the weight for each path sample, and the expected number for each node to be visited.

To calculate the normalization constant $\mathcal{Z}_d$, suppose we have a $L \times L$ lattice and $\tilde{\mathcal{Z}}_n$ for the self-avoid walk of length n, then using resampling, we could generate a collection of SAW(n) using some large sample size N. Then, we count the number of possible directions (up, down, left, right) for each sample and take a summation $\bar{\mathcal{Z}}_{n+1}$. Thus:

$$\tilde{\mathcal{Z}}_{n+1} = \frac{\bar{\mathcal{Z}}_{n+1}\tilde{\mathcal{Z}}_n}{N} \tag{1}$$

Since there are 4 directions for the walker to choose at each step, the number of SAW(n) would grow exponentially. Figure 6 describes the exponentially increasing of Z_n with different dimension D, faster than any polynomials. We should resample after each iteration, otherwise, the sample weight would quickly go to 0, but it would still become less accurate for large walk length n.

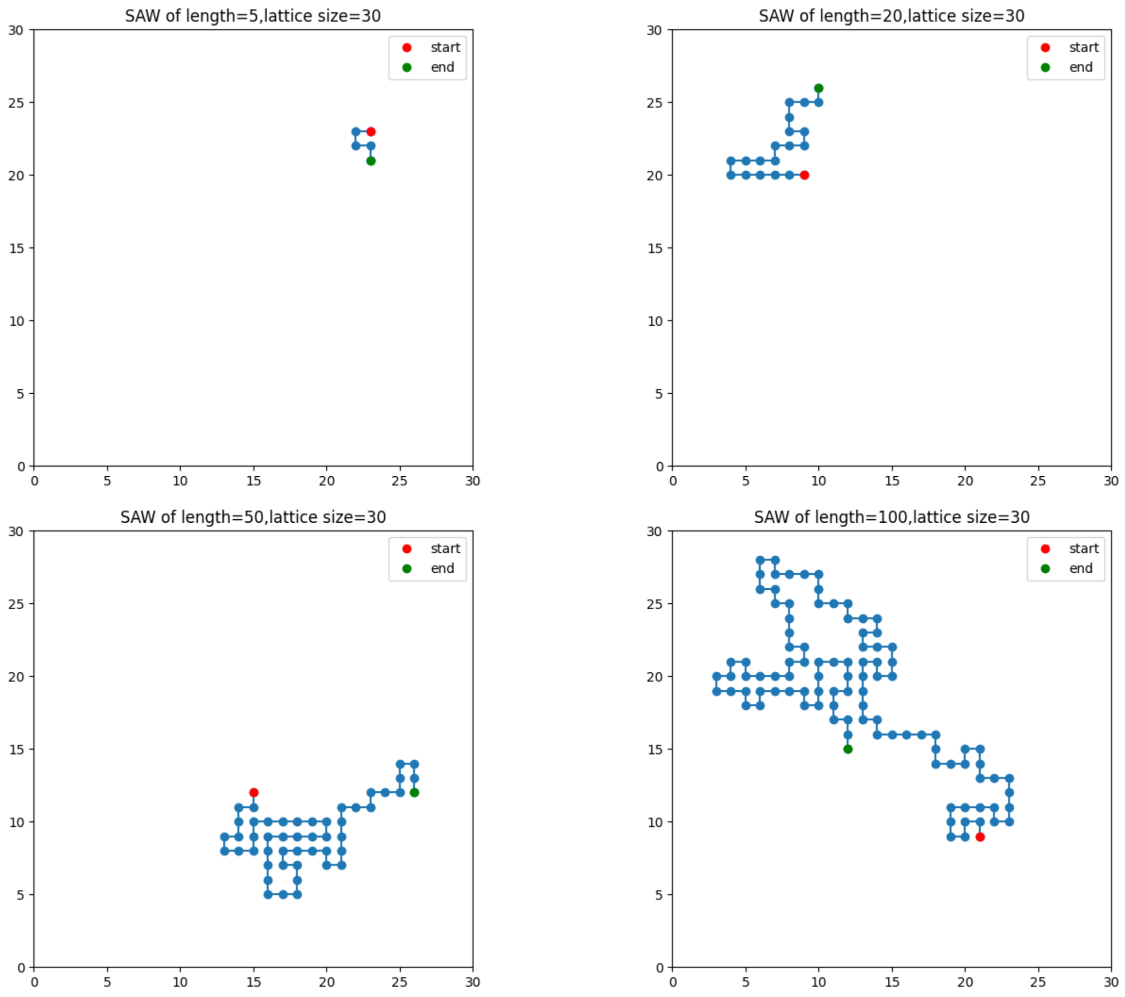


Figure 4: Visualization of sample path of SAW when $D = 5, 20, 50, 100$.

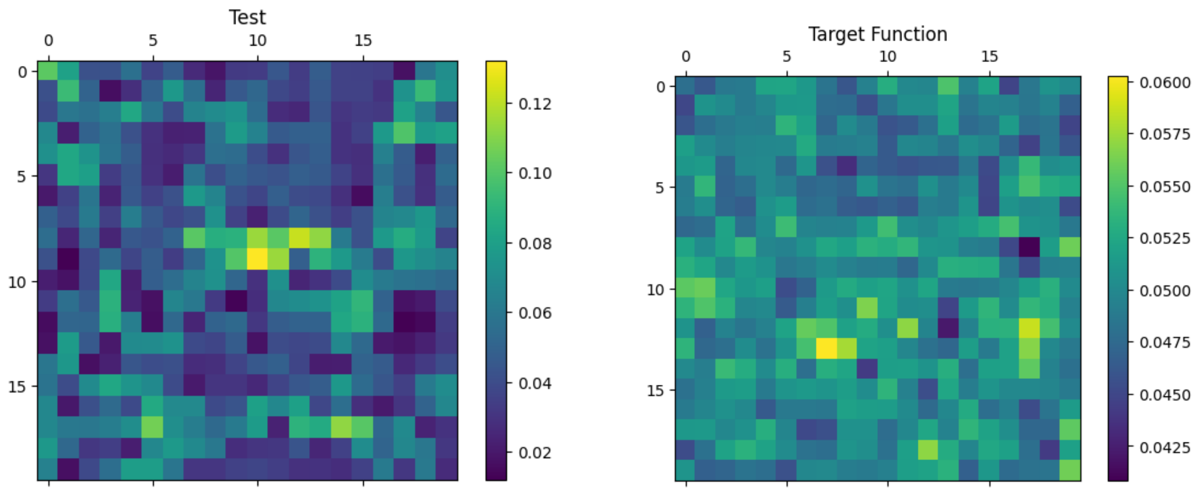


Figure 5: Comparison of frequency chart for without (left) and with (right) resampling.

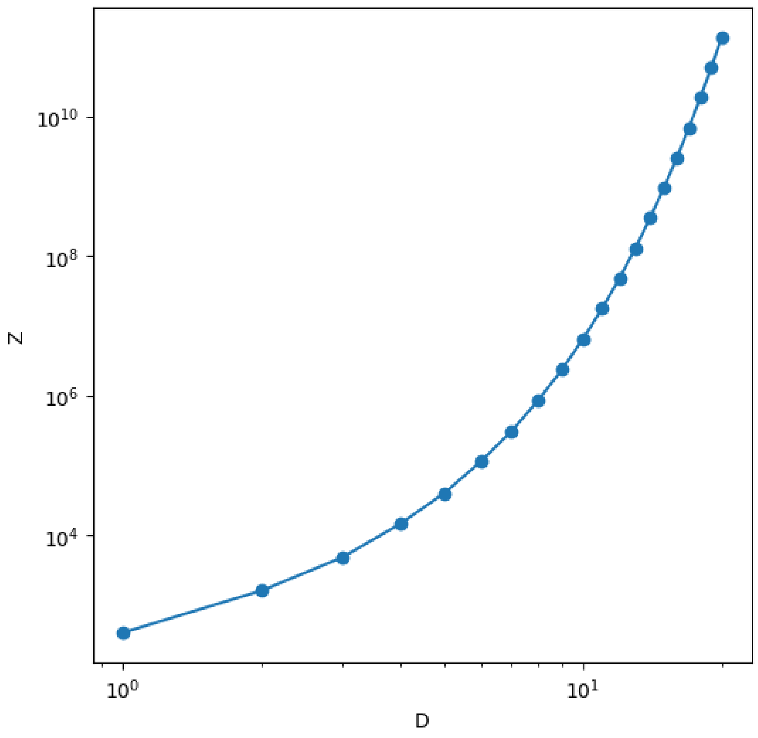


Figure 6: Number of $D-$length SAW in L^2 lattice.